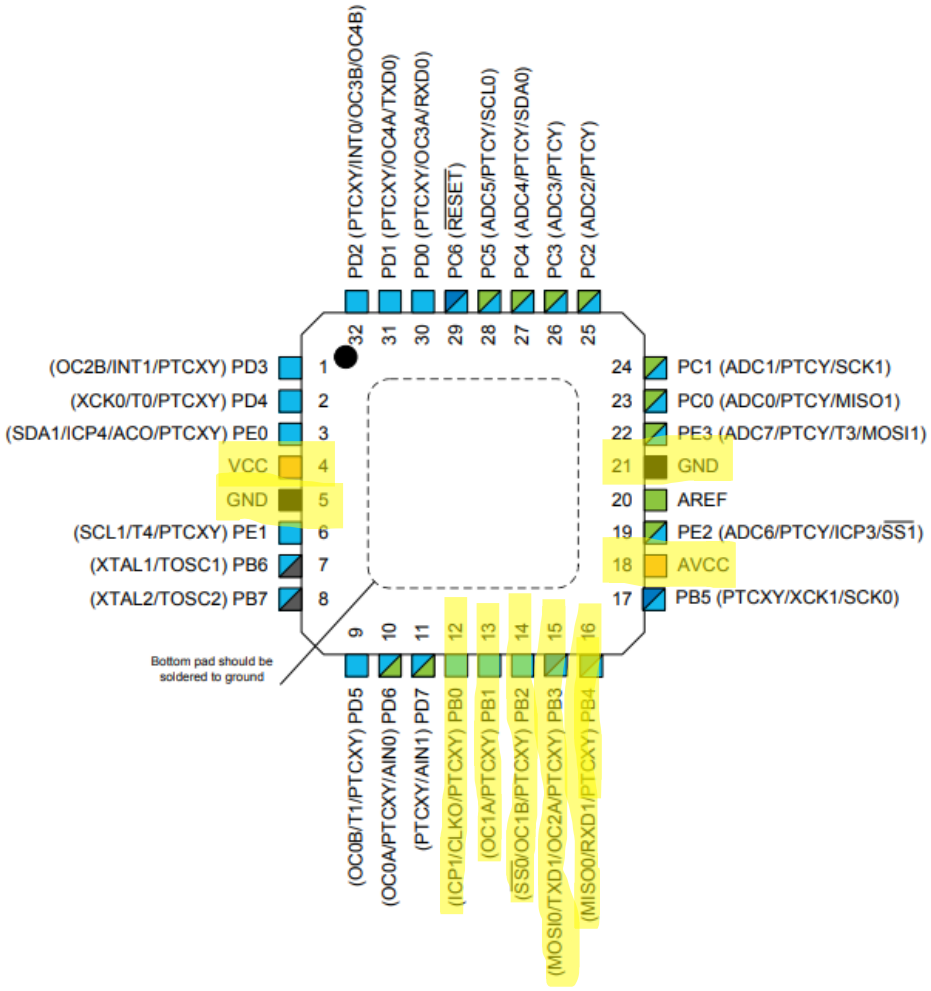
CPE301 – SPRING 2022

Design Assignment 5

Student Name: Ernesto Ibarra-Ayala
Student #: 2001211571
Student Email: ibarre3@unlv.nevada.edu
Primary Github address: https://github.com/ErnestoIbarra333/ErnestoIbarra

1. COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS

Atmel Studio 7.0	Atmega328PB-Xmini PC	Multi-Function Shield	Logic Analyzer
- Assembler	-DC Motor	- Switches	
- Simulator	-Stepper Motor	- LEDs	
- Debugger	-Servo Motor		



2. DEVELOPED MODIFIED CODE OF TASK 1/2/3

```
////////////////////////////////////Task 1:////////////////////////////////////////
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <avr/interrupt.h>

volatile float potentiometer = 0; // value of potentiometer
void adcSetup();
void intSetup();
void PWMSetup();
void readADC();

int main(void)
{
	DDRC &= ~(1<<0); // make portc.0 an input (potentiometer)
	DDRC |= (1<<3); // make portc.3 an output (motor STBY)
	DDRC &= ~(1<<1); // make portc.1 as an input (button)
	DDRB |= (1<<1); // make portb.1 an output (motor PWM)
	PORTC |= (1<<3); // set portc.3 high
	PORTC |= (1<<0); // make portc.0 active high
	PORTC |= (1<<1); // make portc.1 active high

	adcSetup(); // initialize the ADC
	intSetup(); // initialize the interrupt
	PWMSetup();// initialize the PWM

	while (1)
	{
		readADC();
		_delay_ms(100);

		if ((potentiometer >= 62260) && (potentiometer < 65535))
		{
			OCR1A = 62260; // set the PWM to 95% of max
			_delay_ms(50); // wait to set in speed
		}
		else if ((potentiometer < 62257) && (potentiometer >= 3000))
		{
			OCR1A = potentiometer; // set the motor to potentiometer scaled
value
			_delay_ms(50); // wait to set in speed
		}
		else
		{
			OCR1A = 0; //basically turns off motor
		}
	}
}

void adcSetup() //set up the ADC
{
	// use AVCC
	ADMUX |= (0<<REFS1);
	ADMUX |= (1<<REFS0);
```

```
	// select ADC0
	ADMUX |= (0<<MUX2);
	ADMUX |= (0<<MUX1);
	ADMUX |= (0<<MUX0);

	// left align
	ADMUX |= (1<<ADLAR);

	// enable ADC
	ADCSRA |= (1<<ADEN);

	// set pre-scaler to 128
	ADCSRA |= (1<<ADPS2);
	ADCSRA |= (1<<ADPS1);
	ADCSRA |= (1<<ADPS0);
}

void intSetup() //interrupt for pin
{
	PCICR = (1<<PCIE1); // enable pin change interrupt 1
	PCMSK1 = (1<<PCINT9); // Mask for PortC.1
	sei(); // enable global interrupts
}

void PWMSetup() // function to setup the PWM mode
{
	ICR1 = 0XFFFF; // ICR1 as top
	TCCR1A |= (1<<COM1A1)|(1<<COM1B1); // non-inverted mode
	TCCR1A |= (1<<WGM11); // fast PWM
	TCCR1B |= (1<<WGM12)|(1<<WGM13); // fast PWM
	TCCR1B |= (1<<CS10); // start timer
}

void readADC() // function to readADC value
{
	int samples = 15; // number of samples
	potentiometer = 0; // initial potentiometer
	for (int i = 0; i < samples; i++)
	{
		ADCSRA |= (1<<ADSC); // start the ADC conversion
		while(ADCSRA & (1<<ADSC)); // wait until the conversion is done
		potentiometer += ADC; // store the value from the conversion
	}
	potentiometer = potentiometer/15; // take the average value
}

ISR(PCINT1_vect) // timer function
{
	if( (PINC & (1<< PINC1)) == 0 )
	{
		PORTC ^= (1<<3);
	}
}
```

```
//////////////////////////////////Task 2:////////////////////////////////////////
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <avr/interrupt.h>

volatile float potentiometer = 0; // potentiometer value
volatile int moves = 0; //moves
volatile int compValue = 0; // global variable to track number of compValue
volatile int delayValue = 0; // delay

void adcSetup();
void readADC();
void rotate();
void pinSetup();
void ctcSetup();

int main(void)
{
	pinSetup(); // setup functions for movesper/ctc/adc
	ctcSetup();
	adcSetup();

	while (1)
	{
		readADC(); // read pot value

		if (potentiometer > 100)
		{
			delayValue = 1; // set max speed of movesper motor
		}
		else if (potentiometer > 1)
		{
			delayValue = (100 - potentiometer); // scale motor to potentiometer
		}
		else
		{
			delayValue = 10000; // turn off motor
		}
	}
}


void pinSetup() //setups the pins for movesper motor
{

	DDRB |= (1<<1);// set PortB 1-4 as outputs
	DDRB |= (1<<2);
	DDRB |= (1<<3);
	DDRB |= (1<<4);

	PORTB &= ~(1<<1);// start PinB 1-4 at low
	PORTB &= ~(1<<2);
	PORTB &= ~(1<<3);
	PORTB &= ~(1<<4);
}
```

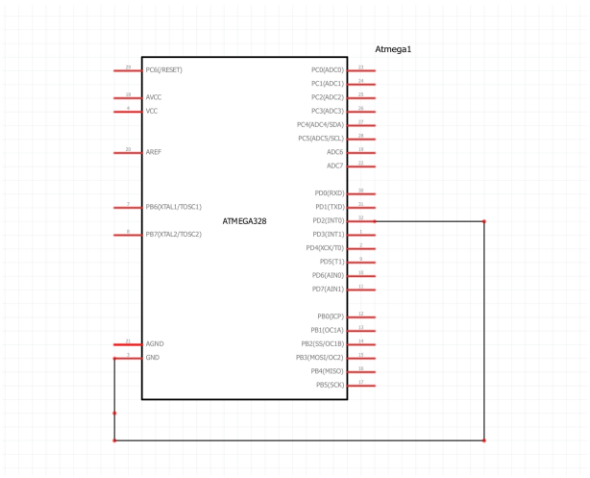
```
void ctcSetup()// function the sets up the CTC Timer
{
	OCR0A = 125; // Set compare register to 125
	TCCR0A = 2; // Enable CTC Mode
	TCCR0B = 4; // set the pre-scaler to 256 and start timer
	TIMSK0 = (1<<OCIE0A); // Enable the timer interrupt
	sei(); // Enable global interrupts
}

void adcSetup() // function to set up the adc
{
	// set PC0 as an input and active high (potentiometer)
	DDRC &= ~(1<<0);
	PORTC |= (1<<0);

	// use AVCC
	ADMUX |= (0<<REFS1);
	ADMUX |= (1<<REFS0);

	// select ADC0
	ADMUX |= (0<<MUX2);
	ADMUX |= (0<<MUX1);
	ADMUX |= (0<<MUX0);

	// left align
	ADMUX |= (1<<ADLAR);

	// enable ADC
	ADCSRA |= (1<<ADEN);

	// set pre-scaler to 128
	ADCSRA |= (1<<ADPS2);
	ADCSRA |= (1<<ADPS1);
	ADCSRA |= (1<<ADPS0);
}

void readADC() // function the reads the potentiometer value using ADC
{
	int samples = 15; // number of samples
	potentiometer = 0; // initial potentiometer
	for (int i = 0; i < samples; i++)
	{
		ADCSRA |= (1<<ADSC); // start the ADC conversion
		while(ADCSRA & (1<<ADSC)); // wait until the conversion is done
		potentiometer += ADC; // store the value from the conversion
	}
	potentiometer = potentiometer/15; // take the average value
	potentiometer = potentiometer/600; // scale the value down by 600 (range of 0-110)
}

void rotate() // here we rotate the movesper motor accordingly
{
	if(moves == 1)
	{
		PORTB |= (1<<1);
		PORTB &= ~(1<<2);
		PORTB &= ~(1<<3);
		PORTB &= ~(1<<4);
```

```
        }
        if(moves == 2)
        {
                PORTB &= ~(1<<1);
                PORTB |= (1<<2);
                PORTB &= ~(1<<3);
                PORTB &= ~(1<<4);
        }
        if(moves == 3)
        {
                PORTB &= ~(1<<1);
                PORTB &= ~(1<<2);
                PORTB |= (1<<3);
                PORTB &= ~(1<<4);
        }
        if(moves == 4)
        {
                PORTB &= ~(1<<1);
                PORTB &= ~(1<<2);
                PORTB &= ~(1<<3);
                PORTB |= (1<<4);
        }
        if (moves == 4)
        {
                moves = 1; //reset to moves 1
        }
        else
        {
                moves++; // else we go to next moves
        }
}

ISR (TIMER0_COMPA_vect) // CTC Interrupt Function
{
        compValue++; // increment compValue
        if (compValue >= delayValue){ // if we delayed long enough (based on potentiometer
value)
                rotate(); // call the rotate function to turn a moves
                compValue = 0; // reset compValue
        }
}



////////////////////////////////////Task 3:////////////////////////////////////////////
#define F_CPU 16000000UL
#include <avr/io.h>
#include <util/delay.h>
#include <stdio.h>
#include <avr/interrupt.h>

volatile float potentiometer = 0; // global variable to hold potentiometer value
void adcSetup();
void pwmSetup();
void readADC();

int main(void)
{
```

```
	adcSetup(); // initialize ADC
	pwmSetup();

	while (1)
	{
		readADC(); // read in pote. value
		if (potentiometer > 570)
		{
			OCR1A=570; // sets the motor to 180 degrees
		}
		else if (potentiometer > 115)
		{
			OCR1A = potentiometer; // set the servo position based on
potentiometer value
		}
		else
		{
			OCR1A = 115; // sets motor to 0 degrees
		}
	}
}

void adcSetup() // Here we set everything to use the potentiometer
{

	DDRC &= ~(1<<0);
	PORTC |= (1<<0);

	// use AVCC
	ADMUX |= (0<<REFS1);
	ADMUX |= (1<<REFS0);

	// select ADC0
	ADMUX |= (0<<MUX2);
	ADMUX |= (0<<MUX1);
	ADMUX |= (0<<MUX0);

	// left align
	ADMUX |= (1<<ADLAR);

	// enable ADC
	ADCSRA |= (1<<ADEN);

	// set pre-scaler to 128
	ADCSRA |= (1<<ADPS2);
	ADCSRA |= (1<<ADPS1);
	ADCSRA |= (1<<ADPS0);
}

void pwmSetup() // function to setup the PWM timer
{
	// setup Timer 1 for non-inverted PWM
	TCCR1A|=(1<<COM1A1)|(1<<COM1B1)|(1<<WGM11);
	// set the pre-scaler to 64 with fast PWM mode
	TCCR1B|=(1<<WGM13)|(1<<WGM12)|(1<<CS11)|(1<<CS10);
	// set the PWM frequency to  50Hz
	ICR1=4999;
	// set PB1 as the PWM output
```

```
	DDRB |= (1<<1);
}

void readADC() // function to read ADC value from the potentiometer
{
	int samples = 15; // number of samples
	potentiometer = 0; // initial potValue
	for (int i = 0; i < samples; i++)
	{
		ADCSRA |= (1<<ADSC); // start the ADC conversion
		while(ADCSRA & (1<<ADSC)); // wait until the conversion is done
		potentiometer += ADC; // store the value from the conversion
	}
	potentiometer = potentiometer/15; // take the average value
	potentiometer = (potentiometer/104) + 115; // scale the range to 125-567
}
```

3. SCHEMATICS

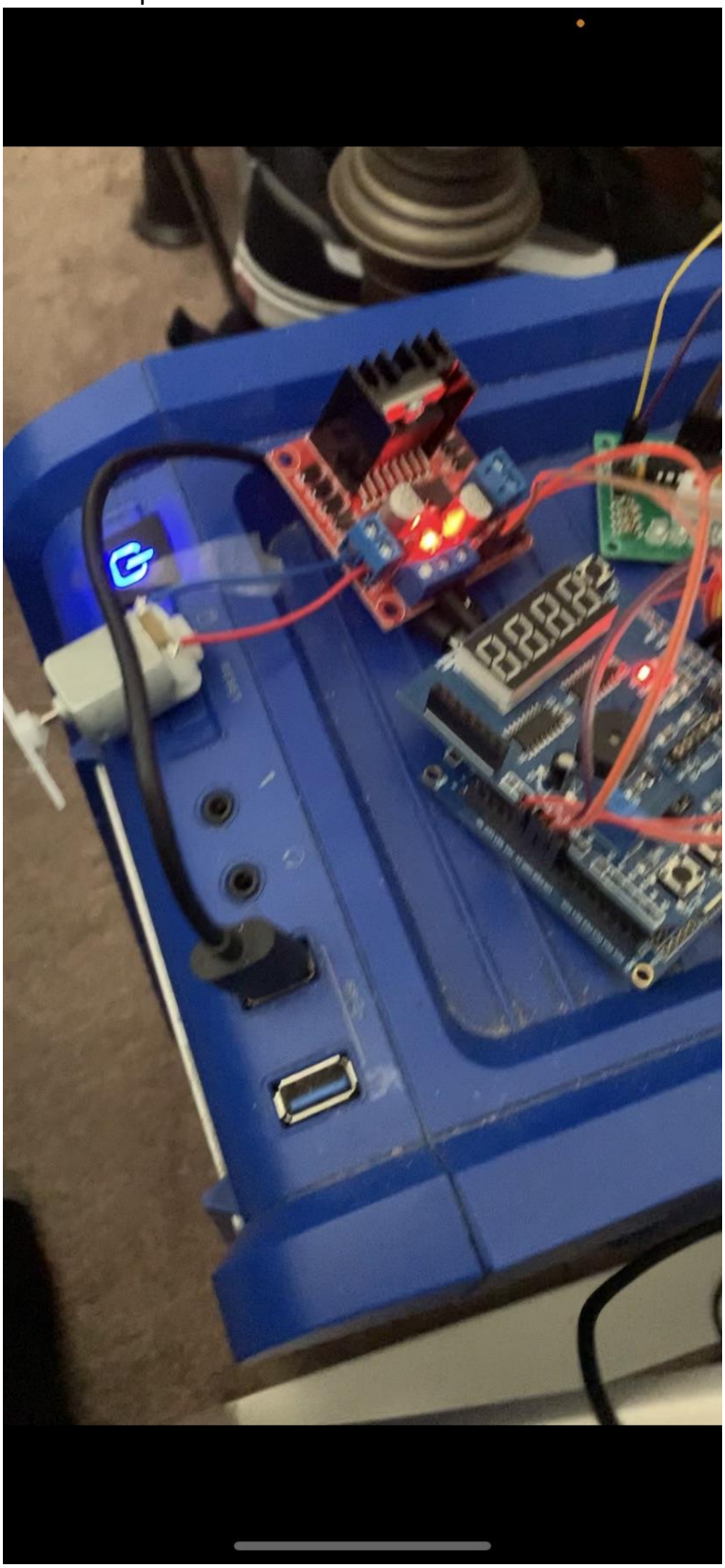


4. SCREENSHOTS OF EACH TASK OUTPUT (ATMEL STUDIO OUTPUT)

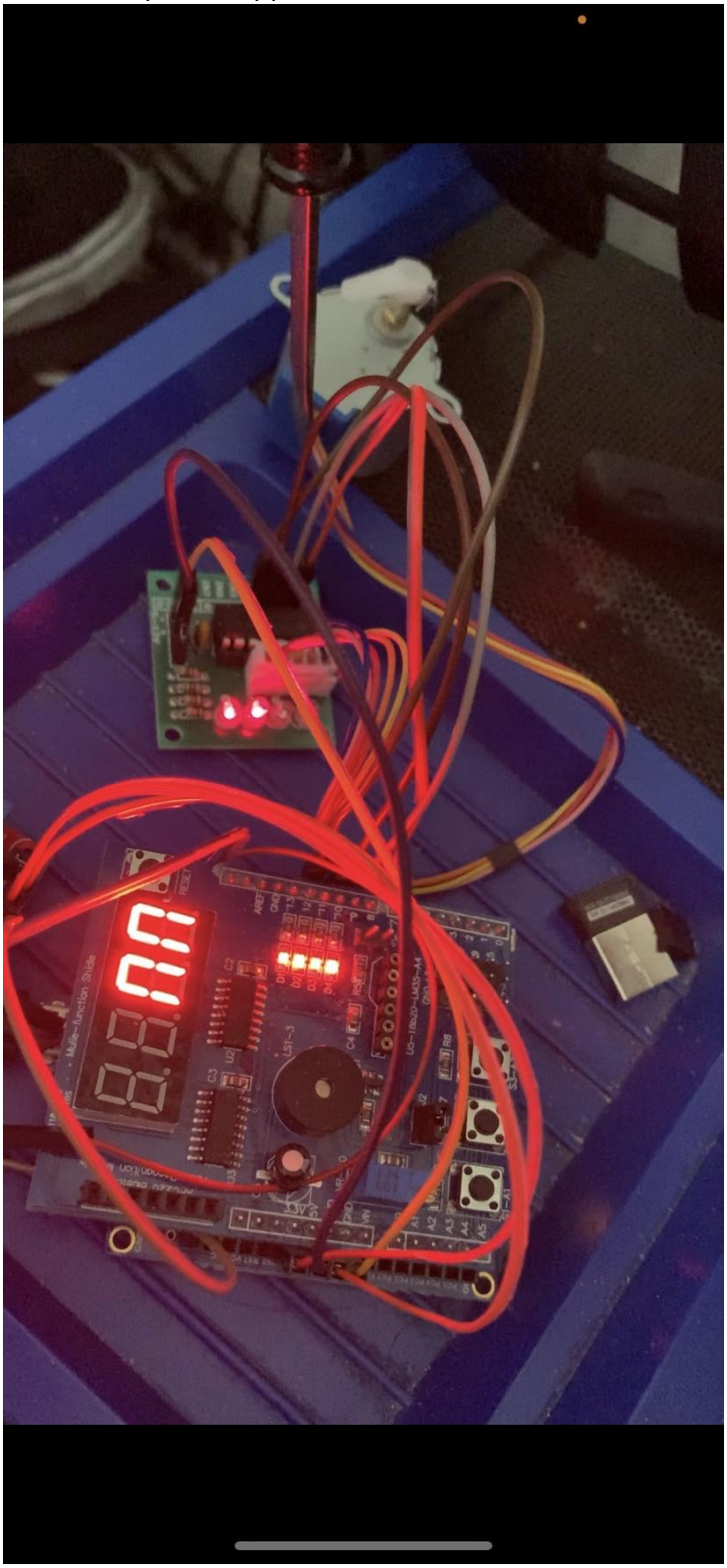
No Task Outputs for this Design Assignment, all on the videos and the board setup.

5. SCREENSHOT OF EACH DEMO (BOARD SETUP)

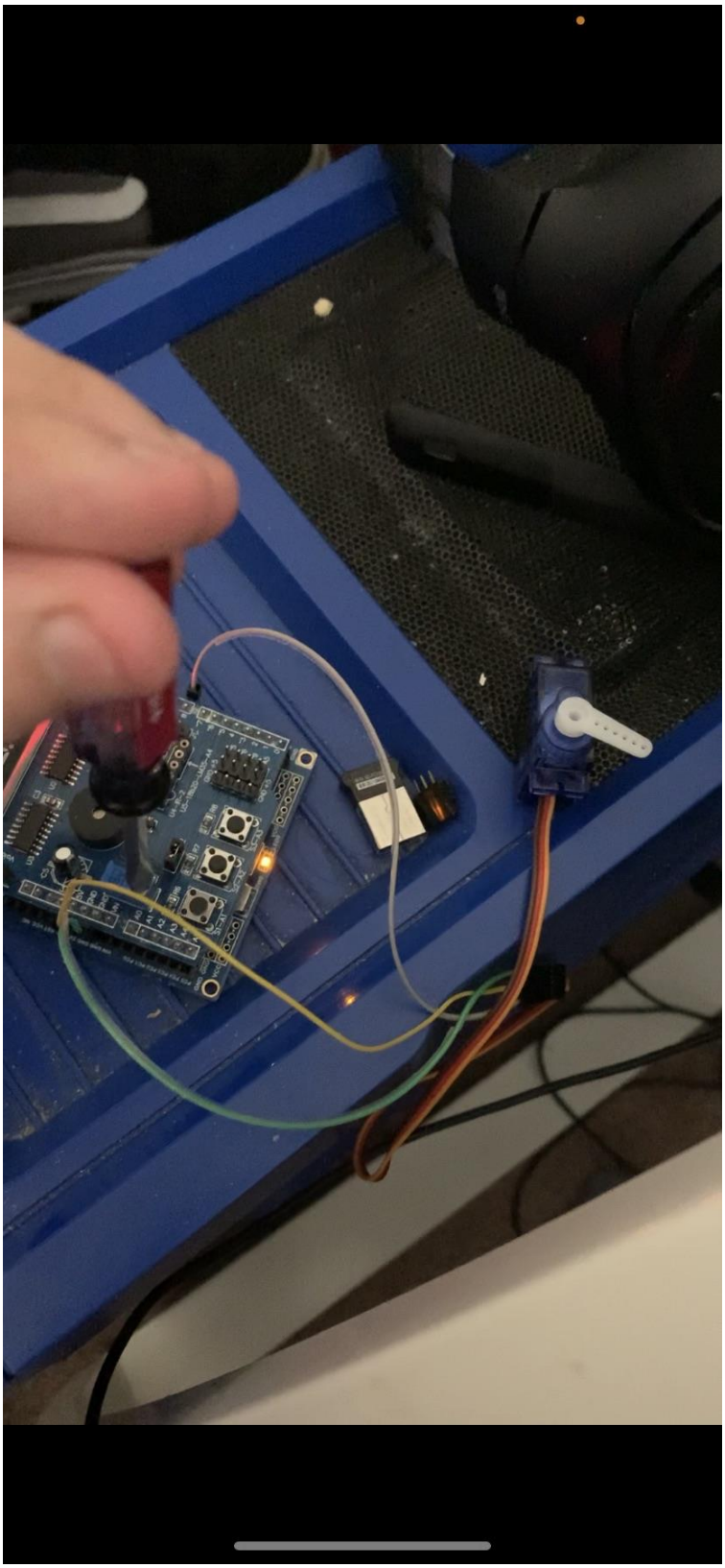
Board Setup for DC motor Task 1.



Board Setup for Stepper motor Task 2.



Board Setup for Servo motor Task 3.



6. VIDEO LINKS OF EACH DEMO

DC Motor: https://youtu.be/IDIETvT6kXg
Stepper Motor: https://youtu.be/y12pRYd9_fM
Servo Motor: https://youtu.be/Y44HIpyd7nw

7. GITHUB LINK OF THIS DA

https://github.com/ErnestoIbarra333/ErnestoIbarra/tree/main/Design%20Assignments

Student Academic Misconduct Policy
http://studentconduct.unlv.edu/misconduct/policy.html

"*This assignment submission is my own, original work*".
Ernesto Ibarra